# Chapter 1

# Gathering What You Need to Develop BlackBerry Apps

### In This Chapter

- Discovering BlackBerry apps and why to develop them
- Collecting the right tools
- Sharpening the right skills
- Meeting the challenges of BlackBerry development

A BlackBerry application is meant to be small, fast, and responsive to its users. BlackBerry smartphones are small, fast, and function as mini-communications centers: a phone, text-messaging system, e-mail client, and Web browser. Your app should give the user the same kind of experience as the standard apps that come with the BlackBerry, providing information quickly and easily with a minimal amount of input.

In this chapter, I show you what tools, skills, and ideas you need to gather and discover to start developing BlackBerry apps.

COPYRIGHTED MATERIAL

## Why Develop BlackBerry Apps?

The BlackBerry App World from Research In Motion (RIM) provides a marketplace devoted to BlackBerry users, and a great many apps of all different kinds have yet to be built. BlackBerry devices have been around a while, mostly as mobile corporate e-mail connections, but the individual consumer is now getting into BlackBerry devices as well, increasing the number of places your app can be running. Figure 1-1 shows the Home Screen of my BlackBerry Curve, with the BlackBerry Browser Application highlighted. Figure 1-2 shows the Browser while running.

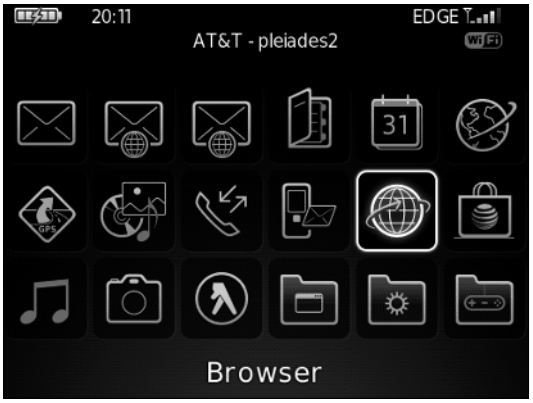


**Figure 1-1:** The BlackBerry Curve Home Screen with the Browser application selected.

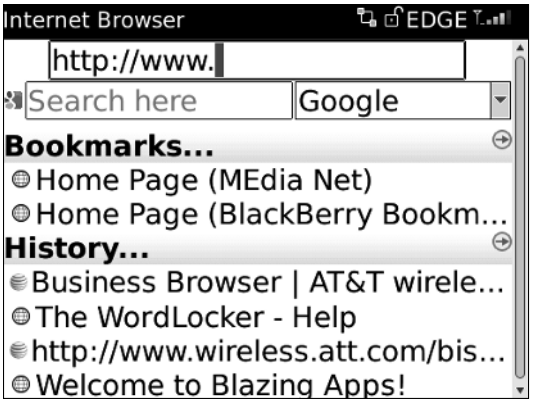


**Figure 1-2:** The Browser application in action.

Here are a few other reasons why I see the BlackBerry as a great development opportunity:

- **BlackBerry applications tend to be small.** This makes them easy to develop and maintain, and they don't require a large development team: You need fewer people to debate the pros and cons of different ways to do the same thing.
- **BlackBerry apps narrowly focus on delivering what the user wants, and no more.** The apps are simple and direct, providing the user with only the information they want — and the tools to get it.
- **BlackBerry apps use Java.** You can leverage any desktop PC Java programming experience you have.
- **The tools and simulators are all free.** You can do all your development on a Windows PC. The simulators all execute the same code as the

actual devices, so you can be sure that if your app works on a simulator, it will work on a real device.

- **The BlackBerry is widely used in corporate enterprises.** From CEOs to administrative assistants and everyone in between, you have a market for business-specific apps that could link everyone in the enterprise.
- **RIM provides the BlackBerry App World for you to showcase and sell your app.** This venue removes the responsibility of credit card handling, hosting, downloading, and notifying users of updates. The App World comes with a variety of pricing tiers, including free and Try & Buy. RIM keeps 20 percent of your application price to cover some of its costs. Submitting your app to the App World incurs a $20 fee per submission, which you can buy in blocks of ten for $200.

# Discovering Apps, BlackBerry Style

BlackBerry users are on-the-go, fast-paced, living in the moment, and your app will need to behave accordingly. BlackBerry users are interested in getting their information *now;* they can't wait more than a few seconds after launching your app to get to the stuff they expect your app to deliver. Your application must accommodate your users and provide them with a means of getting to the value your app adds to their mobile existence. Whether it provides a world traveler with a list of restaurants open around the clock for the city they just arrived in, or merely provides a few moments of entertainment while they're waiting to board their next flight, your app must be easy to use, simple to learn — and, as much as possible, fun.

## Getting familiar with standard apps

The best way to find out more about BlackBerry applications is, well, to use them. Look at the apps that run on a BlackBerry out of the box. You can use a real device or use the BlackBerry simulator that comes with the JDE. Every BlackBerry comes with the following standard apps:

- Browser
- Messages (Email and SMS)
- Contact Manager
- Calendar/Address Book
- Memo Pad
- Tasks

- Calculator
- Alarm Clock

Each one of these applications, written by RIM, contains the basic interactions that BlackBerry users expect to see in your application. BlackBerry users will be using the standard applications often, and so you should become familiar with how users get things done with them. Figure 1-3 shows the BlackBerry Email application as I check my mail account.

**Figure 1-3:** The BlackBerry Email app. BlackBerry Email lets you connect to any e-mail service provider, such as Gmail.

## Understanding how users navigate and use their BlackBerry smartphones

Most users get most of what they need from a BlackBerry application by using just one hand, and often, just by using their thumbs on the trackpad (or trackwheel/trackball for older devices).

The primary input mechanism for a BlackBerry is the pointing device, which can take one of several forms, depending on which model BlackBerry your user has.

- **Trackwheel:** Users move the wheel on the side of the BlackBerry to make the selection highlight move back and forth; users select a highlighted item by pressing in the trackwheel. Figure 1-4 shows a BlackBerry 8700 and its trackwheel.

**Figure 1-4:** The BlackBerry 8700 smartphone provides a trackwheel.

- **Trackball:** Users can move across the pointer around the two-dimensional BlackBerry screen; to select a highlighted item, the user clicks the trackball itself. Figure 1-5 shows a BlackBerry 8830 with its trackball.

**Figure 1-5:** The BlackBerry 8830 smartphone sports a trackball for input.

- **Trackpad:** Users touch the pad lightly to move the selection highlight, and press down to click. You can see the trackpad of the BlackBerry Bold (9700) in Figure 1-6.

**Figure 1-6:** RIM's latest offering: the BlackBerry Bold (9700) and its trackpad.

- **Touchscreen:** Users touch the screen lightly to move the selection highlight, and press down slightly to click the item selected. Figure 1-7 shows you what a BlackBerry Storm looks like, with its touchscreen.

Each input mechanism comes with its own advantages and disadvantages, and this is important when developing your app. You might discover that your app is easy to use on a touchscreen device, yet difficult to use on a device with a

trackwheel. If so, you might decide to create two versions of your app: one optimized for use on a BlackBerry Storm, and one for all the other devices. Keep in mind that you might have to adjust your app based on what type of pointing device the user's BlackBerry supports.

**Figure 1-7:** The BlackBerry Storm with its touchscreen.

Although all BlackBerry devices have a keyboard for users to enter text data, ideally your app should require very little text input. Typing text into a small BlackBerry keyboard is slower than typing on a full-size laptop or desktop keyboard. Investigate whether there is another way for users to provide information.

As long as I'm talking about keyboards, there's one more thing you need to know. BlackBerry smartphones offer two different types of keyboards for users:

- **SureType:** You find this keyboard on the BlackBerry Pearl and Pearl Flip smartphones, as well as a BlackBerry Storm when in portrait mode. Figure 1-8 shows an image of a Pearl with its keyboard. Because the keyboard has fewer keys than a normal keyboard, SureType provides two letters for each key, and the keyboard is laid out like a regular QWERTY keyboard. SureType attempts to predict what a user is typing to speed up the entry of data. Some BlackBerry models with SureType offer a mode called Multitap, where the first tap of a key enters the first letter for that key, and a quick double-tap enters the second letter for that key.

**Figure 1-8:** The BlackBerry Pearl and its SureType keyboard, with its guesses for the word I type.

- **Full:** You find a regular QWERTY keyboard on every other BlackBerry model, as well as on the BlackBerry Storm when in landscape mode. This is the keyboard I prefer to use because all the keys represent one character and I don't have to press a key twice. Figure 1-9 shows the BlackBerry Storm in landscape mode with its full QWERTY keyboard on display.

BlackBerry 9530 Simulator
File Edit View Simulate Tools Help
To:
Cc:
Subject:
Q W E R T Y U I O P
A S D F G H J K L
!?123 Z X C V B N M .
sym space sym
BlackBerry
LCD: (81,244)

**Figure 1-9:** BlackBerry Storm rotated clockwise to show the full keyboard on a touch-screen.

# Filling Your Toolbox

*You* are the most important tool in your software development toolkit. Even when you're working as part of a team, your expertise is more important than the other tools you use to create the code. Your skills in using those tools are what make the tools useful. However, you can't develop BlackBerry apps without the right set of software and hardware tools. The following sections describe the software and hardware you need to gather to start creating BlackBerry apps.

## Downloading the software you need

The following are the major software tools you use to create BlackBerry applications:

- **The BlackBerry Java Development Environment (JDE):** This integrated development environment is available from RIM. The JDE includes the editor, debugger, device simulator, and memory viewer. (See Chapter 2 for the details of choosing a JDE version and downloading it to your PC.)

- **The RAPC compiler:** This is the compiler and linker used by the JDE to produce BlackBerry application files. It makes use of the Sun Java compiler (see the following bullet) to compile your BlackBerry Java code, and then packages it into a form that can be installed onto a BlackBerry device. (The acronym RAPC stands for RIM APplication Compiler, and you don't need to download it — the RAPC compiler comes with the BlackBerry JDE.)
- **The Sun Java compiler:** This is the Sun Microsystems Java Standard Edition (JSE), which must be version 1.5 or later. You can download the Sun Java compiler from

```
http://java.sun.com/javase/downloads/index.jsp
```

I have run into some difficulty using the JDE on 64-bit Windows machines, including Windows Vista and Windows 7. As of this writing, the RIM JDE requires a 32-bit operating system (OS), with a 32-bit version of Sun's Java, in order to run at all. In addition, there is no Macintosh OSX tool for BlackBerry development, unless you use a *virtualization application* (an application that allows you to run other operating systems within it).

RIM offers a plug-in for the Eclipse development environment. *Eclipse* is an open source (free) Java development environment you can download from `www.eclipse.org`. This book concentrates on development using the RIM JDE, but if you're comfortable using Eclipse, you should definitely investigate RIM's plug-in. As of this writing, the current version of BlackBerry Java Plug-in for Eclipse is 1.1, and makes use of the BlackBerry OS 5.0 APIs. Information about the RIM Eclipse plugin can be found at

```
http://na.blackberry.com/eng/developers/devbetasoftware/javaplugin.jsp
```

## Gathering BlackBerry simulators

The BlackBerry JDE comes with several supporting applications to assist you in developing a quality BlackBerry application. You use smartphone simulators to execute your app just as if it were running on a real BlackBerry device. You use the service simulators to represent the real-world services for the BlackBerry to access the Internet (through your PC) or to simulate sending and receiving e-mail. You use simulators in your development process to test your apps before you run them on an actual BlackBerry smartphone. You can download the smartphone simulators from RIM at the following URL:

```
http://na.blackberry.com/eng/developers/resources/simulators.jsp
```

Appendix A contains more information regarding smartphone and service simulators.

Basically, here are the four types of simulator applications you want to use:

- **JDE device simulators:** These come with the JDE, and you launch them with your application already installed. Each JDE comes with its own set of simulated devices, and the newer JDEs have the newest device types simulated. Read more about the device types per JDE version in Appendix A.

  The smartphone simulators that each JDE includes in its set simulates a version of the smartphone OS for that particular version of the JDE. For instance, the BlackBerry JDE 4.5.0 comes with a simulator for a BlackBerry smartphone 8320, and this smartphone shows that it is running smartphone OS 4.5.0.44.

- **Downloaded device simulators:** RIM provides new simulators on its Web site on a regular basis. You can download and install these simulators for free. RIM updates its JDEs less frequently than it releases new devices, so check for new simulator downloads, even while you're in the middle of developing your app. Figure 1-10 shows my application, The Word Locker, running on a simulated BlackBerry Curve (8900).
- **The Mobile Data Service (MDS) simulator:** This comes with the JDE. A BlackBerry device can talk to the Internet only with the help of an MDS. A real BlackBerry will be associated with either the RIM-hosted BlackBerry Internet Service (BIS) or a corporate BlackBerry Enterprise Server (BES). Each of these associations provides MDS services, allowing the device to connect to the Internet. A *simulated* device can't connect to a *real* MDS service, and so the MDS simulator provides Internet access for device simulators. If your application needs to communicate using the Internet, you will have to use the MDS simulator while using a device simulator.
- **The Email Service Simulator (ESS):** This comes with the JDE. BlackBerry users love their e-mail, and RIM has created an e-mail simulator that can act as a gateway to a real e-mail server for a BlackBerry device simulator.

The best thing about all these simulators: They're free!

## Building or buying a development computer

When you're creating apps for the BlackBerry, your choice of computers is limited to a PC running Windows — Windows XP, Windows Vista, and Windows 7 all work with the BlackBerry development tools, but only as long as they are 32-bit versions (not 64-bit versions). Your choice of programming languages is limited to Java, version 1.5 or later, again using the 32-bit version. This pretty much spells out what you need computer-wise to do BlackBerry development. I've worked on several different computers to

do BlackBerry development. In general, it's better to have a fast machine with lots of RAM, though you certainly don't need to buy the most expensive new computer on the market.

**Figure 1-10:** A BlackBerry smartphone simulator running the application WordLocker.

Your development machine will need to handle the following tasks when you're developing BlackBerry software:

- **Editing code:** Any computer running Windows XP can support the requirements of the JDE or any other text editor you prefer. Any machine that can run Windows Vista or Windows 7 is also an appropriate choice.

  Because Microsoft is no longer supporting Windows XP, you might want to use a newer version of Windows. While Windows XP still works well for developing BlackBerry apps, you should consider the advantages of using a newer Windows OS such as Windows 7 because keeping XP running will get more and more difficult as time goes on.

- **Compiling the code into an application:** This is one of the most power-hungry operations you will be performing again and again. Compiling Java code is very CPU- and memory-intensive, so you want a machine that has good processor speed, and as much memory as you can give it.

  As mentioned previously, only 32-bit versions of Java and the Windows operating system can be used for BlackBerry development with RIM's tools.

- **Executing and debugging the application using a simulator:** The device simulators are Windows applications that completely mimic the operations of a real BlackBerry device. The service simulators provide functionality that you use to enable your simulated BlackBerry to access the Internet and send or receive e-mail messages. This requires a machine that has good processor speed and a lot of memory.

A good midrange computer with a large amount of memory, as much as it can use, will support your needs for BlackBerry application development. Table 1-1 shows the requirements for a bare-bones development PC and for a development PC with power to spare.

**Table 1-1 Requirements for a Development PC**

| *Equipment* | *Minimum Requirements* | *Optimum Requirements* |
|---|---|---|
| Processor | 2.0 GHz processor | 3.0 GHz multi-code CPU |
| RAM | 2GB | 4GB |
| Operating system | Windows XP (Service Pack 3) | Windows 7/32-bit |
| Java version | Java 1.5.0 | Java 1.6.0 |
| Network connection | DSL | Cable |

# Choosing a BlackBerry device

The BlackBerry models available as of this writing are

- Bold (9000, 9650, and 9700)
- Curve (83xx, 85xx, and 89xx)
- Pearl and Pearl Flip (81xx and 82xx)
- Storm and Storm2 (95xx)
- Tour (9630)
- 8800 series

These models are available through the major wireless carriers. A particular model may be exclusive to a particular carrier — for instance, the Storm and Storm2 devices are currently Verizon-only, whereas the Bold and Curve devices were originally AT&T-only. AT&T and Verizon strike deals with RIM to be the sole providers of a particular model, but usually the exclusivity is time limited. For instance, Verizon and Sprint have been selling models of the Curve for a couple of years now, after AT&T's contract with RIM to be the only provider selling the Curve ended.

There are pretty much just two ways to acquire a BlackBerry:

- **Purchase one, new or refurbished, from a wireless service provider.** This requires a service contract with the wireless service provider, which might not fit into your budget.
- **Buy one used.** This way is usually less expensive than buying from a wireless service provider but comes with its own set of advantages and disadvantages.

See Appendix B for more information on buying new or used smartphones.

# Using Your Programming Skills

You will need some general skills to develop BlackBerry applications:

- **Java programming:** The Java programming language is the development language for BlackBerry applications. Your source modules must represent Java classes, which get compiled into Java class files and then packaged for the BlackBerry device OS to execute.
- **Debugging:** After you code your app, there's a statistical likelihood that it won't be perfect. If you're lucky, the imperfections will show up rather quickly and obviously. As you develop more applications, you'll find that the obvious and quickly fixed problems happen less often, which leaves the subtle and more-challenging bugs. Your skills at debugging — looking at code as it runs, keeping track of what is going right and what is going wrong, and so on — will play an important part in completing your apps.
- **Software design patterns:** Like with most modern computing platforms, your app will benefit from using software patterns where appropriate. You can certainly create a functional and usable BlackBerry application without relying on any of the canonical design patterns, but applications that are to have a long-duration existence will require a solid structure for operation that the use of software patterns will support. The most obvious is the Model-View-Controller (MVC) pattern, which enforces the separation of your app into pieces that are easy to manage. (See Chapter 4 for more information about MVC.)

# Java programming for BlackBerry

For BlackBerry development, you should be familiar with Java programming in general, and you should know the basics of Java syntax. BlackBerry Java development is somewhat different from desktop PC Java development. The major difference is BlackBerry Java has a somewhat smaller set of classes that you can use to develop applications. Several of the basic Java packages are available:

- `java.io.*`: This package contains the input/output classes you can use to manage retrieving data from and delivering data to various locations, such as Web services and data files.
- `java.lang.*`: You will find the familiar Java base classes such as `String` in this package.
- `java.util.*`: Some of the classes available in the desktop version of Java are available here, such as `Vector`.

Not all the classes available in the JSE version of these packages are available in the BlackBerry version. For instance, the BlackBerry `java.util.*` package consists of only a dozen or so classes and interfaces, whereas the JSE `java.util.*` package comes with almost ten times as many.

RIM removed many classes because of size and performance constraints. For instance, one of the major changes to the Java language was the addition of *generics* in version 1.5, for use with collection classes such as `Vector` (`java.util.Vector`). RIM has not implemented generics for use with BlackBerry development in part because this feature requires a great deal of overhead (such as OS functionality) that doesn't provide a dramatic improvement in the end result in terms of performance.

In addition to these standard Java packages, several of the `javax` and `org` packages are also available:

- `javax.microedition.*`: The Java Micro Edition (JME) packages are all available for you to use. If you've developed a JME application, you can run it without modification on a BlackBerry.
- `javax.bluetooth.*`: Your application can access the Bluetooth hardware on a BlackBerry, if it's available.
- `javax.xml.*`, `org.w3c.dom.*`, `org.xml.*`: These packages provide classes your app can use to read and write blocks of XML data.

RIM provides a fairly rich framework of classes and interfaces that you can use to make your application do just about anything you need. The RIM classes fall into the following three categories:

- **Device interactions:** Your applications use the classes in this category's packages (`net.rim.device.*`) to perform operations that make use of the various parts of the BlackBerry device, such as using the smartphone's GPS or communicating with Bluetooth devices. In addition, the classes you use to create every visual user interface object can be found in this category.
- **BlackBerry application interactions:** You can use the classes in this category (`net.rim.blackberry.*`) to interact with the standard BlackBerry applications. For instance, your app can create an e-mail message and send it, all without the user having to launch the BlackBerry Mail application.
- **Plazmic Media Engine:** This category (`net.rim.plazmic.*`) contains classes your app can use to deliver audio and video content to your users.

## Debugging

The BlackBerry development tools come with a source-level debugger. However, your own skills in debugging Java software are what matter most.

I assume that in your software application development experience, your apps didn't always work perfectly. I'm guessing that you've presumably spent time analyzing code to find where the problems were hiding, and ruthlessly eliminated them. In general, debugging is still something of an art. Sure, tools can assist with finding the place where code goes wrong, but you still need creativity and imagination to know where to start looking. Small applications are usually easy to debug, but when your app has several dozen classes and interfaces, bugs find more places to hide.

You can find resources online and in print regarding debugging, as well as techniques and habits that you can use to make your code easier to debug. I've worked on only a few applications where debugging of some sort was not necessary, and that includes the sample applications I show you in the chapters that follow. All of them had quirks and gotchas that required analysis and imagination to overcome.

## Using software patterns

The world of software has been around long enough that a lot of the ways to solve problems have become standardized. You'll find that using software patterns can greatly simplify your code. This leads to code that's easier to maintain. Patterns tend to be simple and effective, focusing on delivering a limited set of functionality within your app. A class in your application that represents a particular software pattern for achieving a particular objective is straightforward and easy to test.

The simplest example of a software pattern that you might use is one I employ in a great many of my applications: the Singleton pattern. Only one instance of a `singleton` class will exist in an application. Most `singleton` classes have the general appearance shown in Listing 1-1.

**Listing 1-1: The Smallest Form of a Singleton Pattern, Implemented in Java**

```
public class SingletonClass
{
   private static SingletonClass m_instance;

   public static SingletonClass getInstance()
   {
      if (null == m_instance)
      {
         m_instance = new SingletonClass();
      }
      return (m_instance);
}

   private SingletonClass()
   {
      // initialization code
   }

   //
   // the remainder of the methods
   //
}
```

Any code that makes use of this `SingletonClass` will execute `SingletonClass.getInstance()`. This method will instantiate *(create)* and initialize the solitary instance of this class available for the application the first time the method is called, and return that instance that time and every subsequent time the method is called.

Your application can make use of the Singleton pattern when you want to restrict access to one specific location for information. This pattern comes in useful for a large number of different parts of an application, such as

- **User settings:** You will normally have only one user of your application, and so storing that user's preferences for your application in a `singleton` class makes perfect sense.
- **Resource connections:** Access to resources such as a database should be funneled through one object because opening a connection to a resource usually requires substantial code execution. Opening the connection once and maintaining it through your application's lifetime incurs less overhead than multiple openings and closings. You can use a `singleton` class to ensure that a connection is opened only once.

# Understanding BlackBerry Application Development Challenges

If BlackBerry programming were easy, you wouldn't need this book. So you need to be prepared for the inevitable difficulties that will appear. Sometimes these challenges are caused by the BlackBerry device or its OS, and you will have to "code around" these types of problems. Sometimes you will discover you have coded yourself into a corner: for example, a particular decision of how to code something at an earlier moment has forced responsibilities upon your code further down the development path.

The following sections describe the challenges that you'll face in writing BlackBerry applications.

## Choosing an OS version

You have to decide what version of the BlackBerry OS your application will execute on. This is the most significant decision you must make because it will influence and constrain your application's capabilities.

RIM produces new BlackBerry devices — and, therefore, new BlackBerry OS versions — several times per year. In 2009 alone, the following new devices were released:

- Storm2
- Bold 9700
- Curve 8900
- Tour

Each of these devices came with its own new version of the BlackBerry OS. OS versions are usually represented by four numbers separated with dots, such as 4.7.0.113. The first two numbers are usually the most important; they are the major and minor version numbers. The difference in behavior between OS 4.6.0.49 and OS 4.6.0.75 is likely to be minor although I have come across times when such a version upgrade has fixed a bug that my code encountered.

It's true! Sometimes you will discover that your perfectly debugged and packaged code comes across a bug in the OS!

The differences between minor OS versions — say, OS 4.6 versus OS 4.7 — will likely be significant. Of the two just mentioned, OS 4.7 is interesting because it's the only OS that runs on the BlackBerry Storm and Storm2 devices. What's interesting about these devices? Two things:

- Storm and Storm2 use the touchscreen as their primary input mechanism. This permits your app to make use of users sliding their fingers around the screen, enabling your app to perform some actions more fluidly.
- Storm and Storm2 have accelerometers to indicate the orientation of the display. Your app can take advantage of knowing whether the user has switched the device's orientation from portrait (taller than wide) to landscape (wider than tall). You can see the Storm showing its landscape mode in Figure 1-9.

Each JDE has a version number as well, and this version number corresponds to the version number of the OS that you can code for. I can't stress this fact enough:

*The version of the JDE you choose will limit which devices your app can run on.*

For more information about choosing a JDE version, see Chapter 2.

The JDE version is the minimum device OS version that your app will execute on. Your application's code will usually be *forward compatible:* That is, newer device OS versions will usually execute code created using an earlier JDE version. This isn't always true, but it's a safe bet for the next several subsequent versions of the OS. This also depends on what BlackBerry classes your app uses to do its job. A bare-bones minimalist application that does something simple such as take user text input and store it in memory isn't likely to run afoul of changes in the device OS for several versions to come.

RIM releases a device OS about once every year that runs on all of a set of the currently available devices. As of this writing, a version of OS 5.0 can run on all the devices that were released in 2009, plus several others that came out in 2008. Currently, the OS version that runs on the greatest number of smartphones is OS 4.5.0.

Using version 5.0 of the JDE, you can develop applications that will run on every device using OS 5.0. This will mean your app can use classes that take advantage of the features listed above for the Storm series of devices but still run (without any ill effects) on non-Storm devices. However, because a non-Storm phone (such as a BlackBerry Pearl Flip) has neither a touchscreen nor an accelerometer, code written to take advantage of these features won't install on the device because the OS won't know what to do when it comes across those specific features in your app. However, a Pearl Flip running OS 5.0 will allow code implemented to use those specific features to execute, although that part of the code will simply do nothing.

The BlackBerry App World, RIM's online marketplace for BlackBerry applications, allows you to deploy separate apps that differ only based on which device each one is intended to run on. RIM recognizes that you want your application to run on as many different devices as can access the App World, but that you also want your application to take advantage of the features each device provides, instead of just coding to one common-denominator, lower-level device OS.

Thus, your application can exist in the App World in multiple forms. For example, if you intend for your app to be used only by those users who have accelerometers in their BlackBerry devices, you can restrict your app to an OS 4.7-only zone, and no other device will be able to install it. ***Note:*** If you want your app to run on all devices but also take advantage of accelerometers when running on a Storm, you have to develop two versions of your app: one for Storm devices and then one for all the rest.

The key point is that these two different applications only constitute one app submission to the App World, so you're only paying for one submission instead of two. Chapter 12 goes over the details you need to know about the App World, and your app's life there.

## Programming defensively

Your application hopefully will be used by tens of thousands of people worldwide. When that happens, each user becomes a stress test for your app. These users will inadvertently discover ways of breaking your application — causing unforeseen consequences to occur — that you never thought of. Your users will be your next major challenge.

Users follow a bell curve in terms of their use of your app. Most will behave exactly as you expect, following a "happy path" of operation where they never encounter shortcomings in your app and never try to make the app do something incorrect. But a small faction of users either intentionally or unintentionally drives your app into a situation it's not prepared to handle. If you're working for a large organization with a Quality Assurance department, a great many of these situations can be discovered and resolved before your code hits the outside world. If you're a solo entrepreneur, the burden falls on your shoulders.

Most abnormal situations are a result of invalid user input. This specific cause can be eliminated through the use of fixed-input entry fields: UI elements that restrict user input to particular data types or specific values. In addition, your app should "sanitize" the data that users can provide as input, to make sure that nothing bad gets inside your app to wreak havoc.

For instance, your app may want users to enter a date value, perhaps for a birthday reminder. You could use a simple text-editing component, such as an `EditField`, for users to select and start typing in a date. But you would then have to make sure that what they enter is actually a real date — this could be any of the following:

- 09/01/10
- September 1, 2010
- 2010-09-01

As you can see, letting users enter arbitrary text data forces you to implement the appropriate conversion method for turning their entries into a form more suitable for your app to make comparisons and calculations with. However, a basic text-entry field also allows your users to enter text such as *Karl's Birthday* just as easily.

This text will pretty much crash your application if it tries to add or subtract days, months, or years to or from it. A better way to ensure that only valid date values, easily converted to a usable form, are entered is to use UI components that *limit* what a user is permitted to enter. In this case, using a component such as a `DateField` would be much more appropriate. Unless you want to give your users more flexibility — which forces you to convert every possible input to an appropriate, usable form within your app.

Another potential cause of problems is users who download a version of your app that is not valid for their smartphone OS. Luckily, the BlackBerry App World gives you the ability to provide multiple versions of your application customized to the different smartphones that RIM supports. You discover all this in Chapter 12.

# Entering a Brave, New BlackBerry App World

With the success of the Apple App Store for delivering software to iPhone devices, major smartphone manufacturers and even some wireless service providers are designing and deploying their own storefronts to sell applications designed for their platforms. As mentioned earlier, RIM hosts the BlackBerry App World, which is an online marketplace for all kinds of BlackBerry applications. The App World provides many categories of applications for BlackBerry users to download, such as Business, Education, Games, News, Shopping, and Utilities. (For the complete list, see Chapter 12).

Each category is further subdivided, allowing prospective buyers to drill down through the store and find the app that's right for them. You, as a developer, should become familiar with the user's experience of searching and finding apps in the BlackBerry App World, to place your app in the right category and to make sure your app shows off its best face. You find out about all the App World categories and their subcategories in Chapter 12.

## Deciding what kind of app to create

If you already have an idea for an app, great! You've passed one of the most difficult steps on your way to BlackBerry App World riches! Figuring out what

you want to develop can be challenging, so I recommend that you review the apps available in the App World in a variety of different categories. Think of this exercise as window shopping. Your imagination will be working in the background while you look at what's for sale already, and all it takes is one example to trigger something wonderful.

## Brainstorming, alone or in groups

I am a co-founder of a small startup, BlazingApps. The other co-founders and I sit down irregularly for a brainstorming session. We go around the table and contribute an idea or many about different apps we've thought about since the previous meeting. I enjoy this kind of imaginative collaboration and highly recommend it. Granted, you might find it difficult to brainstorm all by yourself, so for the solo entrepreneur, I highly recommend writing down any thoughts or ideas somewhere so that you can easily find them again. You should write down everything, no matter how small or unimportant you might think it is. This way, you'll have a collection of thoughts and imaginings that you can use to spark more of the same. Or else, some combination of your thoughts and imaginings across a spectrum of your recorded notes will group together and prove to be an app worth creating.

### How one idea leads to another

For instance, in a project on another smartphone device, I once had a need to d isplay text in a variety of the available fonts to demonstrate to the user experience specialist what the device's different fonts would look like. Several apps were available at this platform's marketplace to do this, and even writing my own was fairly straightforward.

But it got me thinking: The fundamental problem I was trying to solve was an inability on the part of the development team to quickly and clearly see what the results of our choices for user interface elements would actually end up looking like. There was no way for the user experience specialist to "play around" with different settings and get an instant reaction to the settings on a real device.

Unless. . . . What if I created an app that allowed a user of the smartphone to mix and match fonts and buttons and labels and menus and all the other user interface objects on the phone itself? In essence, this would be an "interface creator" application, running on the actual device, so that anyone with a device could "try out" different combinations of user interface items. Developing such an application would be helpful to anyone in an organization who wants to prototype user interface development, without actually having to write code.

This is the kind of imagination-behind-the-scenes that can lead to applications that your users will want to use. I haven't gone out to build this particular app, yet, but you have my permission to make the attempt.

Even if you're a solo entrepreneur, talking with other developers or even just acquaintances can generate ideas. You can scan the BlackBerry online forums to pick up what issues users are running into and develop ideas based on problems they are encountering and — obviously — want a solution for.

## Becoming a BlackBerry developer and App World vendor

Your first step toward BlackBerry app development is to register with RIM as a BlackBerry developer. You then gain access to a treasure trove of all things BlackBerry, including

- All BlackBerry JDE downloads
- All BlackBerry simulator downloads
- Articles pertaining to BlackBerry development (the Knowledge Base)
- The official BlackBerry online development forums
- All developer documentation

The developer registration process is pretty straightforward. I go over the steps involved in Chapter 2.

Becoming a BlackBerry App World vendor is a little more complicated, with more steps to follow because (you guessed it) money is involved. You need a PayPal account. I go over these details as well in Chapter 2.